Software Development

Higher Diploma in Science in Computing

Project-Based-Learning Project Report

Report Authors in Alphabetical Order:

Liz Bourke/19149310

Alexia Falcoz/[]

Laurine Rolland/[]

Contents

[Coding Responsibilities 3](#_Toc26125305)

[Input, Processing, and Output 4](#_Toc26125306)

[Class Diagram 5](#_Toc26125307)

[Decisions Made 6](#_Toc26125308)

# Coding Responsibilities

Every member of the project team undertook to code the entire project prior to discussing implementable solutions, to ensure that everyone understood the principles at work. Based on the group discussions, we decided to divide the coding responsibilities for the final product in the following manner:

**Alexia** took primary responsibility for coding solutions to store the game history, for implementing the arrays, and for checking that all numbers generated by the Random Number Generator (the lottery numbers) and all numbers input by the user are unique values.

**Laurine** took primary responsibility for coding the instantiable class, its setters and getters, and the function to calculate winnings.

**Liz** took primary responsibility for coding repetition and selection statements inside the application class.

# Input, Processing, and Output

## Input

There are two main inputs provided by the user:

1. The number of lines (between one (1) and three (3)) that the user plays in a given game, and
2. The six (6) numeric values between one (1) and forty (40) which the user provides for each individual line.[[1]](#footnote-1)

There is one main input that must be generated at the beginning of each game:

1. The six (6) random numeric values between one (1) and forty (40) against which the user-generated values must be checked.

## Processing

There are five (5) main pieces of processing to be accomplished:

1. For each line, the user-generated numeric values must be checked against the random numbers, and the number of matches recorded.
2. For each line, the number of matches recorded must be checked against the number of matches needed to win a sum of money (3, 4, and 5 matches) or the whole lottery (6 matches).
3. For each game, the sum of money won must be calculated and recorded.
4. For the total of all games, the total sum of money won must be calculated.
5. For the total of all games, the average sum of money won per game must be calculated.

## Output

There are 6 main outputs:

For each game, the following must be output to the user:

1. the total number of matched numeric values per line played;
2. the number of lines played in the game;
3. the number of lines where the user won a sum of money. (That is, the number of lines where the user matched between 3 and 6 numbers.)

At the end of all the games, the following must be output to the user:

1. the total winnings. (The sum of all sums won in each game.);
2. the total number of games played;
3. the average winnings per game.

# Class Diagram

[Laurine, you're up]

# Decisions Made

Decision 1: We chose to store the number of lines per game and the numeric values input by the user for each line as a 2D array, in order to simplify processing. The other option we considered was to:

1. use a variable to count the number of lines
2. take the numeric values as a 1D array
3. calculate the number of matches in that line
4. store the number of matches at index[i][linecounter] of a 2D ArrayList, where the outer ArrayList index[i] was equal to the number of the current game.[[2]](#footnote-2)
5. increment the linecounter variable
6. take the numeric values again
7. repeat until there are no more lines to calculate matches for.

But this approach meant that we were running three calculations to find matches-per-line every game. (Also, clarification of the project specification indicated that use of ArrayLists was deprecated.) Using a 2D array, we could run a (larger) calculation just once.

Decision 2: We chose to limit the array that holds game history to a theoretical maximum of 100 games, on the basis that this lottery game as a whole is not fascinating enough that anyone would play it more times than that in a single session.

Decision 3: We decided that the lottery jackpot would be a total of 1,000,000 euro. Anyone who "wins the lottery" (matches 6 numbers in a line) will receive 1,000,000 euro.

Decision 4: We decided that winning the lottery would automatically end the game loop (because why reward greed?) and trigger the display of games history.

1. We chose to store this as a 2D array: see "Decisions." [↑](#footnote-ref-1)
2. This approach would mean we could have a theoretically infinite number of games. [↑](#footnote-ref-2)